1 Introduction

This period focused on establishing the core communication infrastructure and the primary control loop for lane keeping. The team underwent a member substitution, requiring a redistribution of tasks to maintain the development pace of the autonomous driving stack.

2 Planned activities

- Infrastructure (Setup & Connectivity): Finalize the configuration of the Raspberry Pi 5 and establish a robust remote access protocol via VNC to facilitate headless development.
  - Type: Environment preparation. Lead: Team Leader.
- Perception (Lane Detection): Research and develop a computer vision pipeline focused on edge detection (Canny/Sobel) to extract lane boundaries from camera frames in real-time.
  - Type: Research & Development. Lead: Computer Vision.
- Control (Lateral Guidance - Stanley Implementation): Design and implement a Stanley Controller algorithm to minimize the cross-track error (e_y) and heading error (ψ) relative to the detected lane and send the data to the Nucleo board.

$$\delta(t) = \theta_e(t) + \text{atan}\left(\frac{k \cdot e_y(t)}{v(t)+ks}\right)$$

  - Type: Development. Lead: Control.
- Simulation (Virtual Twin): Install a localized Linux environment, configuring ROS and the Gazebo simulator to validate control laws before hardware deployment.
  - Type: Testing. Lead: Simulation.
- Hardware (Validation Track): Construct a physical test track with standardized lane markings to validate the integration of the vision thread with the steering actuators.
  - Type: Environment preparation. Lead: Hardware/Integration.

3 Status of planned activities

- **Infrastructure** (Completed): Successfully configured the Raspberry Pi 5 with remote access via VNC and a correct dashboard setup. Difficulties:
  - Library errors during the setup.sh (specifically with the versions flask and pyhton socket io) and error communication between the Frontend and Backend via sockets, requiring a rewrite of the message-handling protocol and dashboard src.
- **Perception** (Completed): The system extracts lane boundaries and calculates cross-track/heading errors. The pipeline stabilizes camera input via grayscale and Gaussian filtering, utilizes Canny/Sobel operators for robust edge detection under variable lighting, and applies dynamic ROI masking to streamline processing. Difficulties:
  - The edge detection filters were initially highly sensitive to ambient light variations, requiring constant threshold recalibration to avoid losing lane makers in high-contrast areas.
  - Processing high-resolution frames in real time exceed the Raspberry Pi 5's constraints which forced the team to optimize the pipeline using a specialized ROI a dawn-sampling techniques.

  - Glossy track surfaces created “ghost edges” due to reflections necessitating the addition of morphological operations to clean the binary mask before calculating the errors.
- **Control** (Ongoing - 80%): The Stanley control thread is fully integrated operational. Successfully transitioned from mathematical modeling to active PWM command dispatch to Nucleo board. Currently performing fine-tuning of the controll. Difficulties:
  - Integration challenges with the messageHandlerSender and messageHandlerSubscriber functions when interfacing with the car's low-level communication stack.
- **Simulation** (Completed): We completed a full Gazebo setup. Created a structured catkin workspace. Developed a custom shell script to automate the sourcing of ROS variables and the local environment. Difficulties:
  - Resolved camera data flow issues by re-configuring the /automobile/ namespace in model.sdf, fixed traffic_light_pkg compilation via CMakeLists.txt adjustments, and identified an oversized, black-textured "Stop" sign, although it has not obstacled our current progress our current progress.
- **Hardware and track** (Completed): Constructed a physical test track for real-world validation of the steering actuators. Difficulties:
  - Lab’s space is not enough for making the whole test track, in the future the team will have to use a bigger space.

4 General status of the project

Currently, our car has successfully transitioned from a basic manual startup to a functional autonomous lane-keeping state.

- The vehicle can autonomously follow lane boundaries using real-time edge-detection vision algorithms processed on the Raspberry Pi 5.
- It maintains lateral guidance through a Stanley Controller, which minimizes cross-track and heading errors during straight paths and moderate curves.
- The development stack is mirrored in a ROS/Gazebo environment, allowing for rapid testing of control laws before physical implementation.
- The system currently encounters instability (oscillations) at higher speeds due to the ongoing syntonisation of controller gains.

5 Upcoming activities

- Perform fine-tuning of the Stanley Controller gains to eliminate lateral oscillations and ensure smooth navigation at varying speeds.
- Develop and implement scripts for obstacle detection using LiDAR data to enable basic collision avoidance and environmental mapping.
- Integrate the perception and control threads to signs, intersections, parking, roundabouts and other elements
- Finalize the construction of the test track with the necessary requirements and establish a stable connection between the car and competition servers.